Refactored Characteristics of Intelligent Computing Systems

Christopher Landauer, Kirstie L. Bellman

Aerospace Integration Science Center, The Aerospace Corporation

cal@aero.org, bellman@aero.org

Abstract

We have discussed the following measurable characteristics of intelligent behavior in computing systems: (1) speed and scope of adaptibility to unforeseen situations; (2) rate of effective learning of observations; (3) accurate modeling and prediction of the relevant external environment; (4) speed and clarity of problem identification and formulation; (5) effective association and evaluation of disparate information; (6) identification of more important assumptions and prerequisites; (7) creation and use of symbolic language.

In this paper, we isolate some common underlying capabilities for these characteristics, and show how they can all be produced using those capabilities. We describe the architecture of a system that has all of these underlying capabilities, using our Wrapping integration infrastructure to coordinate and organize a large collection of models and other computational resources. In particular, these models include complete models of the system's resources and processing strategies, and therefore a model of its own behavior, which it can use to affect that behavior.

Key Phrases: Computationally Reflective Infrastructure, Constructed Complex Systems, Intelligent Autonomous Systems, Knowledge-Based Polymorphism, Layers of Symbol Systems, Problem Posing Interpretation

1 Introduction

In an earlier paper [28], we described some important characteristics of intelligent computing systems, and discussed how to make measurements of those characteristics for performance assessment. In this paper, we find some common themes underlying our earlier list of capabilities, and show how the new list of more fundamental characteristics can be used to implement the more operational capabilities. This "refactoring" is an example of our principle of *layers of symbol systems* [8] [27] [29], which reflects our observation that biological systems overlay functionality in complex ways, at least partly due to their opportunistic exploitation of side effects. These overlays do not remove the lower level functionalities; they merely supersede them in appropriate contexts.

Intelligence is difficult to measure, because it is thought to be an intrinsic property of systems, like a potential capability or competence, whereas the only things that can be measured are actual performances under various kinds of condi-tions. This problem has plagued the evaluators of human intelligence since the beginning, to the point that they have generally concentrated on measuring some postulated corresponding performance characteristics [13].

In our opinion, intelligence metrics can only be based on observed system behavior (though the observations can, of course, measure internal processes from an internal perspective, since we often have some kinds of internal access). Success in a particular task is not by itself the right criterion. Measuring performance to infer competence, even of externally observable behavior, is problematic, since we cannot make the measurements over a sufficiently wide range of situations.

Similarly, the intelligence or even genius that invents and develops good models for physical phenomena is not always adequately (or even correctly) assessed by fitting the theory to the data: Copernicus' model of the solar system was much less accurate than Ptolemy's for many years after it was first formulated; the latter fit the observations far better. It wasn't until Kepler decided to use ellipses instead of circles, and Brahe provided much more and much more accurate data, that the sun-centered model became dominant. This very long-term, partially social, comparison and evaluation process is not how we want to assess the intelligence of systems that we build.

2 Autonomy, Intelligence, Creativity

Intelligence is much more mundane than programs playing hard symbolic games or proving theorems. It is also much more remarkable. The ability for an organism to exist in the world, to move around, interact with its environment, and make experience-dependent decisions, is one of the deepest mysteries of life. These things that most living things can do (at least for a while) are much more interesting and difficult than the things only humans can do, and the things most humans can do "without thinking" are much more interesting and difficult than the things a few humans can only do after much study or training.

We view intelligence as sitting in a collection of related qualities that include autonomy and creativity. We take autonomy to be a kind of lower bound for intelligence and creativity to be a kind of upper bound, without actually assuming that any of them implies the other. We frequently use postulated or speculative explanations of autonomous, intelligent, and cre-

Report Documentation Page

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
AUG 2002		**00-00-2002 to 00-00-2002**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Refactored Characteristics of Intelligent Computing Systems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
The Aerospace Corporation,Aerospace Integration Science Center,Los Angeles,CA,90009	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Proceedings of the 2002 Performance Metrics for Intelligent Systems Workshop (PerMIS ?02), Gaithersburg, MD on August 13-15, 2002

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	**Same as Report (SAR)**	**8**	
unclassified	**unclassified**	**unclassified**			

ative human behavior as a guide to what can be done, and as a source for our study of what computer programs can be made to do.

There are really only two classes of requirements for effective autonomy (both are difficult): robustness and timeliness. Robustness means graceful degradation in increasingly hostile environments, which to us *implies* a requirement for adaptability, and timeliness means that situations are recognized "well enough" and "soon enough", and that "good enough" actions are taken "soon enough". There is never any *optimization* here, since it takes too much time and produces non-robust solutions.

In the previous paper [28], we concentrated on the measurement problem instead of the construction problem, though we mentioned some definite ideas about how to build these interesting programs, based on our Wrapping infrastructure for Constructed Complex Systems [18] [22] [24]. We discuss some of them in more detail in Section 7.

3 Computing System Behaviors

We make the assumption that a computing system is designed to help its users _do_ something [14]. That something is a problem in some subject area, such as, for example, copy a file in a computer system, produce a document in a legal office, kill monsters and collect treasures in a computer game, retrieve a web page for a user, solve an equation in a mathematical subject area, find patterns in noisy data in a scientific field, coordinate a distributed simulation for a military application, launch a spacecraft in the aerospace business, collaborate remotely on a design problem for space systems, etc..

In all of these cases, there is an *application domain*, which provides a certain context of use and corresponding terminology. Actually, this is more of a *domain-specific language*, since it includes more than just vocabulary terms. It also has a set of abbreviations and conventions about what can remain implicit, and a set of simplifications (which are fruitful lies about the entities and behaviors in the domain). It is important to note that these languages might or might not be written symbolically, since, for example, a computer game is often commanded using a joystick instead of typed commands, and some immersive Virtual Environments are commanded by user movement and gesture (and eventually speech).

What the user wants to do is called the *problem*, which only makes sense within the context of interpretation provided by the domain-specific language of the application domain. These languages are used to define the problem context or *problem space*, which is a specialized context within the application domain, in which it makes sense to state a problem.

In other words, it is our opinion that a problem cannot be even stated properly or sensibly without an agreed upon (more often, merely assumed) application domain and problem context. Very often, it is mistakes in the common understanding of this problem context that leads to unexpectedly bizarre or constricting behaviors on the part of the computing system.

So now we have a well-specified problem defined in a problem context. We are purposely setting aside creativity for now, though we believe that this framework can also be applied in that case, with a problem statement of finding the appropriate well-defined problem (this approach is part of our *Problem Posing* paradigm [21]). Explicitly identifying the problem, and separating it from the possible solutions or required user actions, is an important aspect of our approach. It allows many different possible solution methods to be considered. Since NO one analysis or problem-solving method can deal with all problems in a complex domain [12] [43] [7], it is important to have many methods available.

These form the *resource space*, which contains the computational and information resources that are available to address the problem. It is usually implemented as a large set of independent methods, but we think that more structure here can help (which is why we call it a space).

A certain configuration of those resources is needed to address the particular problem that the user has specified. This collection is usually much smaller than the total resource space, so we call it the *solution space*. Since it contains only those resources required to solve the problem, we would ideally like to have the computing system find this space quickly.

However, in order to find a solution space, very often a much larger *examination space* or *discovery space* must be searched.

For example, in trying to prove a theorem (in geometry, say), the problem space is one in which the assertion can be made, the solution space is one in which the proof can be made, and which often involves extra elements constructed just for the proof. The resource space is the collection of lemmas, theorems, inference rules, problem-solving methods, and previously solved problems, and the solution search space is much wider, since it has to include many different kinds of construction and proof discovery methods.

4 Characteristics of Intelligence

In this Section, we list the measurable characteristics of intelligent systems from [28], and discuss what they require and how they might be implemented. We expand the characteristics into their parts and organize them, so we can find a more basic list of aspects that can produce these ones. We may or may not be able to measure these underlying aspects, but we intend to use them to get some notion of how independent the characteristics are. It is clear to us that each of the characteristics above can be implemented using many of the other characteristics at a lower level of detail. These details will lead us to an architecture proposal in Section 7. In the next Section, we show how they can be refactored:

1. speed and scope of *adaptibility* to unforeseen situations,

2. rate of effective *learning*,

3. accurate modeling and *prediction* of the relevant external environment,

4. speed and clarity of *problem identification* and formulation,

5. effective *association* and evaluation of disparate information,

6. identification of *assumptions* and prerequisites, and

7. creation and use of *symbolic language*.

We make no claim that these are all the important characteristics; discovering others is the point of our research program in this area.

By far the most commonly expressed attribute of intelligence is *adaptibility*, which for us means the speed and scope of adaptibility to unforeseen situations, including recognition (of the unforeseen situation), assessment, proposals (for reacting to it), selection (of an activity), and execution. Accurate prediction of effects is even better (and more successful), but we save that one for a later Section.

To *adapt*, a system needs many flexibilities, ways of deciding that change is needed, and ways to effect that change. In particular, a very large number of computational resources, many ways to use them (i.e., many different ways to map problems into applications of them), and a strong ability to decide how to use them.

Another common attribute of intelligence is *learning*, which for us is the rate of effective learning of observations, behavior patterns, facts, tools, methods, etc. [37].

To *learn*, a system needs to be able to abstract a situation into a representation of it, to identify the important parts, and to change its processes to improve the performance. This improvement process is more than adjusting performance towards an evaluation criterion, it is also at least partly about inventing the appropriate criteria.

An important way to be less surprised at environmental phenomena is *predictive modeling*, which for us means accurate modeling and prediction of the relevant external environment. This kind of modeling includes the ability to make more effective abstractions (which is treated below in a later Section). Since a system cannot know everything about its environment, we assume that there will be multiple models carried in parallel, with new data interpreted into information using the model as an interpretive context, and each model adjusted, assessed, and ranked for likelihood continually.

To build *models*, a system needs to be able to identify the important features and processes of a phenomenon in an environment, and to represent them in an explicit model in a conceptual space. It also needs to be able to compute abstractions and evaluate their appropriateness.

The best way to respond to problems quickly is to *identify* them quickly, which requires speed and clarity of problem identification and formulation. In our opinion, speed of problem solution is secondary. Even if a problem is specified as a constrained search, humans seem to construct search spaces and algorithms that are very problem-specific, often extremely intricate, constructed using the constraints directly (i.e., not by searching a large encompassing space, and masking or otherwise ignoring the parts outside the constraints).

To *identify problems* (and situations), a system needs to be able to abstract the situation into some explicit representation, to compare it to previously identified or encountered problems and situations, and determine which aspects of those earlier situations are important for or relevant to the current one.

One of the clearest signs of intelligence is the wide scope and effectiveness of *associations*, and the corresponding evaluation of disparate information for inclusion into a decision process. Discovery and explanation of new associations is even frequently associated with creativity.

To compute *associations*, a system needs to have appropriate representations of phenomena in many conceptual spaces, and processes that examine and compare elements in those spaces. These comparison processes underlie all grouping and clustering of elements.

A perennial problem with reasoning in systems, and particularly with deduction, is the mis-identification and conflation of *assumptions*. It is important that a system can identify its more important assumptions and prerequisites, which includes the ability to widen a context (by removing some of the assumptions).

To identify *assumptions*, a system needs to be able to examine its own reasoning processes, infer missing assumptions from coverage failures in the implicit context, and discover gaps in what can be represented and how it can be processed. This process of discovering what is missing is one of the hardest ones, since it requires the system to have some kind of notion of entities and processes outside its own representational systems.

Perhaps the most important property of all, in our opinion, is the use of *symbolic language* for explicit representations, including the range and use of analogies and metaphors (this is about identification of similarities), and the invention of symbolic language, which includes creating effective notations for internal representation. This property is not altogether unchallenged, but despite the "behavior-based" intelligence work [38], we believe representation to be essential at all levels of autonomy and intelligence [8], especially for computing systems.

To use symbol systems and *symbolic methods*, a system needs to base its entire processing on explicit representations, and at the same time be able to change those representations in part or entirely, based on internal assessments of success or failure.

It is for this reason that we want these systems to have complete models of their own processing [35], which we have shown is not only not very hard, but is actually fairly straightforward when we allow multiple component resources that can

apply to the same problems in appropriate contexts. These self-models are directly connected to the behavior of the system, either through compilation or interpretation, so that changing the models changes the behavior. The models may be written using *wrex* [21] [23], together with any other notations that the system can interpret.

This use of the Problem Posing interpretation of programming languages leads to what we call a *Problem Posing Interpretation* of behavior: all behavior is applying computational resources to posed problems, and while biological systems generally do not have explicit access to those problems, we expect our Constructed Complex Systems to be more informed. They will have an active knowledge representation using *wrex*, and they will use partial evaluation for situation-dependent local code improvement [23]. They will use Computational Reflection to monitor and adjust their own behavior, because they will be able to examine and alter their own representations.

5 New Abstractions - Refactoring

In this Section, we describe a few properties that are common in all of these characteristics, which we will identify as enabling capabilities.

First and foremost is identification of important aspects of a situation, which depends on the representational scheme in use at the time. This process is a kind of focussing of attention that is essential for any kind of intelligent behavior [2] [40].

The next important process is the determination of whether or not a situation is a problem (and if so, what problem it is). This process involves comparison to earlier situations, simulation of consequences of internal models of the current situation, and assessment of their relative likelihood.

Next comes representing the situation in some collection of conceptual spaces, using modeling spaces and model construction methods that allow models with variables that can be specified later. This requires computational access to a very large class of conceptual spaces, as well as to a variety of analyses within and among them.

An important part of limiting the consideration is deciding what to ignore, that is, assessing the importance or relevance of aspects of the representation, as well as that of the outstanding problems at hand.

All of these process imply the ability to retain large numbers of these models, with multiplicitous indexing that allows access in several different ways.

Finally, the system will need continual contemplation of the models, their efficacy, the symbol systems in which they are defined, and their efficacy, in order to assess its own adequacy and the occasional need for re-expression.

To summarize, there are a number of different kinds of computational resources that are needed to provide the properties we have listed:

- methods of abstraction and simplification

- methods of evaluation in context

- methods of representation (conceptual spaces)

- methods of retention and retrieval

- methods of comparison and association

In addition, we require mappings from problems to resources in context, as well as explicit representations of everything (the contexts and problems, all of the resources, and the processes, including the mappings and the reasoning methods). Then the entire system is grounded in its symbol system definitions and assumptions, which are themselves explicit, so that they can be changed.

6 Intelligent Systems

These issues affect the design of Constructed Complex Systems [17], which are artificially constructed systems that are managed or mediated by computing systems, since it is clear that they will not have the same kind of experience in the real world that makes biological systems so remarkable. The success or otherwise of the interaction between them and their human users will determine the usefulness of these systems, and therefore of these measurements.

A human using a computing system of any kind is presented with a Virtual World, that is, an environment in which the user is allowed to perform some limited set of control actions, and is presented with some limited set of information displays. The number and variety of available control actions is almost always very small, and only occasionally determinable. The number and utility of information displays is almost always not enough. These appallingly limited worlds are so restrictive in their scope and so poor in their quality of interaction that using them proficiently becomes an exercise in excessive focus on certain details, and can only be performed successfully by a few people [36]. We would like to build systems that are much more helpful.

This problem with inflexibility is why we are concerned with issues of autonomous and intelligent behavior in such systems, which for us, at least means that the system takes a major role in selecting and evaluating its own goals [18] [3] [15] [27]. When we expect Constructed Complex Systems to operate autonomously, whether out in the real world or in cyberspace, we need to incorporate a great deal of flexibility and adaptability into their design and implementation. We have shown one way to implement such a system [22] [24], one that also helps avoid the most common difficulties found in complex computing systems: rigidity and brittleness.

7 System Architecture

Biological systems have much more flexible and powerful adaptation properties than most constructed systems [11], and

a careful consideration of their properties provides stringent requirements for the kind of Constructed Complex Systems that would be able to act autonomously. It also gives us some hints about the design structures that are needed [39] [18] [34].

Our approach is to define a new kind of architecture [24] [31] [32], based on our studies of "Integration Science" [9] [10]. It includes our Wrapping integration infrastructure [23], which uses Knowledge-based Polymorphism to provide a great deal of flexibility of component interaction, our Problem Posing interpretation [21], which is a declarative interpretation of all programming languages, so that posed problems can be separated from applicable resources, and our conceptual categories [16] [26] [30] to provide a flexible representation mechanism that separates model structures from the roles they play. These design choices allow our systems to make their own abstractions [33], and construct their own models.

Our Wrapping architecture provides the required flexibility by supporting systems that are variable as far down as we choose to make them (even all the way down through the operating system to the hardware) [41] [17]. One reason that we want this variability is that we expect to study many different approaches to any given problem area, and our infrastructure has to support alternatives for almost every part of every process. In fact, one of the principles we have highlighted in our architecture investigations is that NO one model, language, or method suffices for a complex system (or environment), so the variability is not just convenient; it is necessary [12] [43] [7] [10].

In addition, we take the hypothesized common origin of language and movement processes [8] as a hint, since the implied layers of symbol systems can be implemented easily in Constructed Complex Systems using a meta-level architecture with Wrappings [18] [22] [24] [27].

In addition to the data and processes, we also need a third style of computation, that of "re-expression", which allows a system to re-organize itself when its current organization is not adequate. What this means for us is that the system can somehow detect when its own representational mechanisms are not adequate, and it can use the failures to help invent new ones.

To make things even more interesting, we also want to have the system decide for itself when it needs to be re-organized, because its fundamental symbol systems are not expressive or powerful enough, and then carry out for itself the re-organization automatically, by defining new symbol systems and re-expressing itself in the new terms. This behavior is hard to implement usefully, but we have made some progress in identifying the important issues. In particular, the "get stuck" theorems show that any Constructed Complex System, faced with a complex external environment, will have to analyze and invent its own symbols and representational mechanisms [20] [25].

The Wrapping processes give the process structure and the Wrappings and conceptual categories give the data structure. The re-expression criteria are implemented as resources that monitor the system. We describe each of these technical issues in turn, and then show how they can be used to help construct the kind of system we want to build.

The essence of computation is interpretation of symbol systems. The only operations that a digital computer can perform are copying and comparison. All arithmetic in digital computers is via limited-precision explicit models of the corresponding integer or real arithmetic. Therefore, we cannot construct computing systems to do complex or otherwise interesting tasks without many implicit or explicit models of the kinds of computation, deduction, or analysis required. All of these models must then be expressed in terms of the operations that we can implement on these (very) limited computers. It is far too easy to become constrained by the expressive mechanisms used [41], and part of our approach to intelligent computing systems reduces that dependence by making the symbol system and representational mechanisms variable.

The theorems of Turing, Gödel, and others show that there are fundamental limits on the expressive and computational power of computing systems, but ALL of the theorems assume that the symbol system remains fixed (that is a basic assumption in all of the mathematical proofs), and that the parallelism can be mapped into interleaved events. Systems that are not restricted in either of these ways might escape the bounds of these theorems. This study is one of our current directions of research in Computational Semiotics [20] [24] [25].

There are several fundamental mathematical questions involved in this study: (1) how self-reference can be made not only possible but sensibly computable (using the new self-referential set theory [1] [5] [6]), (2) how formal mathematical structures can be extended to incorporate more information about context (using situation theory [4] to capture the context of behaviors [42]), (3) how to move these formal structures into new contexts and assess the resulting validity (using our conceptual categories [16]), (4) how to define mathematical structures before the basic elements are defined (also using conceptual categories), (5) how to capture more of the modeling process in mathematical structures (using both situation theory and conceptual categories), (6) how to decide when a notational system is inadequate (using internally computable evaluation criteria [20]), and (7) how to fix it.

These questions are part of our New Math Initiative [19], since we believe that some of these questions require new kinds of mathematics, or at least new applications of mathematics, instead of the "heroic engineering" and other systematic methods that we have relied on thus far for building Constructed Complex Systems.

8 Conclusions

We care about measuring intelligence because we want to engineer devices that have some of those characteristics, and without some better measurement processes, we will have no re-

peatable way to evaluate and compare different designs.

We have described some properties that we think are important, that have driven our research in Constructed Complex Systems, including a few that have not been extensively used or identified in the literature. We do not think that they completely cover the spectrum of what is commonly considered to be intelligent behavior, but they do cover more of the scope than simply "adaptability" or "intellect".

We have identified some fundamental enabling capabilities that seem to underlie these properties, and described an architecture that includes all of these enablers, as a way to test our assertions about the connection between them and intelligent behavior. We expect that as we build systems with more of these enablers, the systems will exhibit more of the important properties we have identified, and at the same time they will seem more intelligent.

We think that this problem is hard, and that we are on a right track (we make no assumption about how many right tracks there may be; the more we collectively explore, the more likely it is that we will get some of the right answers). We think that fundamental investigations like these are necessary; we hope that they are sufficient.

References

[1] Peter Aczel, *Non-well-founded Sets*, CSLI Lecture Notes #14, Center for the Study of Language and Information, Stanford U., U. Chicago Press (1988)

[2] James S. Albus, Alexander M. Meystel, *Engineering of Mind: An Introduction to the Science of Intelligent Systems*, Wiley (2001)

[3] K. S. Barber, C. E. Martin, "Agent Autonomy: Specification, Measurement, and Dynamic Adjustment", in Henry Hexmoor (ed.), *Proceedings of Agents'99/ACS'99: Workshop on Autonomy Control Software*, 1 May 1999, Seattle, Washington (1999)

[4] Jon Barwise, *The Situation in Logic*, CSLI Lecture Notes No. 17, Center for the Study of Language and Information, Stanford U. (1989)

[5] Jon Barwise, John Etchemendy, *The Liar: An Essay on Truth and Circularity*, Oxford U. (1987)

[6] Jon Barwise, Lawrence Moss, *Vicious Circles*, CSLI Lecture Notes No. 60, Center for the Study of Language and Information, Stanford U. (1996)

[7] Kirstie L. Bellman, "An Approach to Integrating and Creating Flexible Software Environments Supporting the Design of Complex Systems", pp. 1101-1105 in *Proceedings of WSC'91: The 1991 Winter Simulation Conference*, 8-11 December 1991, Phoenix, Arizona (1991); revised version in Kirstie L. Bellman, Christopher Landauer, "Flexible Software Environments Supporting the Design of Complex Systems", *Proceedings of the Artificial Intelligence in Logistics Meeting*, 8-10 March 1993, Williamsburg, Virginia, American Defense Preparedness Association (1993)

[8] Kirstie L. Bellman and Lou Goldberg, "Common Origin of Linguistic and Movement Abilities", *American Journal of Physiology*, Volume 246, pp. R915-R921 (1984)

[9] Kirstie L. Bellman, Christopher Landauer, "Integration Science is More Than Putting Pieces Together", in *Proceedings of the 2000 IEEE Aerospace Conference (CD)*, 18-25 March 2000, Big Sky, Montana (2000)

[10] Kirstie L. Bellman, Christopher Landauer, "Towards an Integration Science: The Influence of Richard Bellman on our Research", *Journal of Mathematical Analysis and Applications*, Volume 249, Number 1, pp. 3-31 (2000)

[11] Kirstie L. Bellman and Donald O. Walter, "Biological Processing", *American Journal of Physiology*, Volume 246, pp. R860-R867 (1984)

[12] Richard Bellman, P. Brock, "On the concepts of a problem and problem-solving", *American Mathematical Monthly*, Volume 67, pp. 119-134 (1960)

[13] K. Anders Ericsson, Reid Hastie, "Contemporary Approaches to the Study of Thinking and Problem Solving", Chapter 2, pp. 37-79 in Robert J. Sternberg (ed.), *Thinking and Problem Solving*, Academic Press (1994)

[14] Kenneth D. Forbus, Johann de Kleer, *Building Problem Solvers*, A Bradford Book, MIT Press (1993)

[15] Catriona M. Kennedy, "Distributed Reflective Architectures for Adjustable Autonomy", in David Kortenkamp, Gregory Dorais, Karen L. Myers (eds.), *Proceedings of IJCAI-99 Workshop on Adjustable Autonomy Systems*, 1 August 1999, Stockholm, Sweden (1999)

[16] Christopher Landauer, "Process Modeling with Conceptual Categories", Paper dtkmi01 in *Proceedings of HICSS'00: The 33rd Hawaii International Conference on System Sciences (CD)*, *Track II: Decision Technologies for Management*, *Modeling Knowledge-Intensive Processes Mini-Track*, 4-7 January 2000, Maui, Hawaii (2000)

[17] Christopher Landauer, Kirstie L. Bellman, "Constructed Complex Systems: Issues, Architectures and Wrappings", pp. 233-238 in *Proceedings of EMCSR'96: Thirteenth European Meeting on Cybernetics and Systems Research*, *Symposium on Complex Systems Analysis and Design*, 9-12 April 1996, Vienna, Austria (April 1996)

[18] Christopher Landauer, Kirstie L. Bellman, "Computational Embodiment: Constructing Autonomous Software Systems", *Cybernetics and Systems: An International Journal*, Volume 30, Number 2, pp. 131-168 (1999)

[19] Christopher Landauer, Kirstie L. Bellman, "Where's the Math? The Need for New Mathematical Foundations for Constructed Complex Systems", in *Proceedings ICC'98: the 15th International Congress on Cybernetics*, 23-27 August 1998, Namur, Belgium (1998)

[20] Christopher Landauer, Kirstie L. Bellman, "Situation Assessment via Computational Semiotics", pp. 712-717 in *Proceedings of ISAS'98: the 1998 International MultiDisciplinary Conference on Intelligent Systems and Semiotics*, 14-17 September 1998, NIST, Gaithersburg, Maryland (1998)

[21] Christopher Landauer, Kirstie L. Bellman, "Problem Posing Interpretation of Programming Languages", Paper etecc07 in *Proceedings of HICSS'99: The 32nd Hawaii Conference on System Sciences, Track III: Emerging Technologies, Engineering Complex Computing Systems Mini-Track*, 5-8 January 1999, Maui, Hawaii (1999)

[22] Christopher Landauer, Kirstie L. Bellman, "Computational Embodiment: Agents as Constructed Complex Systems", Chapter 11, pp. 301-322 in Kerstin Dautenhahn (ed.), *Human Cognition and Social Agent Technology*, Benjamins (2000)

[23] Christopher Landauer, Kirstie L. Bellman, "Generic Programming, Partial Evaluation, and a New Programming Paradigm", Chapter 8, pp. 108-154 in Gene McGuire (ed.), *Software Process Improvement*, Idea Group Publishing (1999)

[24] Christopher Landauer, Kirstie L. Bellman, "Architectures for Embodied Intelligence", pp. 215-220 in *Proceedings of ANNIE'99: 1999 Artificial Neural Nets and Industrial Engineering, Special Track on Bizarre Systems*, 7-10 November 1999, St. Louis, Mo. (1999)

[25] Christopher Landauer, Kirstie L. Bellman, "Symbol Systems in Constructed Complex Systems", pp. 191-197 in *Proceedings of ISIC/ISAS'99: International Symposium on Intelligent Control*, 15-17 September 1999, Cambridge, Massachusetts (1999)

[26] Christopher Landauer, Kirstie L. Bellman, "Relationships and Actions in Conceptual Categories", pp. 59-72 in G. Stumme (Ed.), *Working with Conceptual Structures - Contributions to ICCS 2000, Auxiliary Proceedings of ICCS'2000: The 8th International Conference on Conceptual Structures*, 14-18 August 2000, Darmstadt, Shaker Verlag, Aachen (August 2000)

[27] Christopher Landauer, Kirstie L. Bellman, "Reflective Infrastructure for Autonomous Systems", pp. 671-676, Volume 2 in *Proceedings of EMCSR'2000: The 15th European Meeting on Cybernetics and Systems Research, Symposium on Autonomy Control: Lessons from the Emotional*, 25-28 April 2000, Vienna (April 2000)

[28] Christopher Landauer, Kirstie L. Bellman, "Some Measurable Characteristics of Intelligence", Paper WP 1.7.5, *Proceedings of SMC'2000: The 2000 IEEE International Conference on Systems, Man, and Cybernetics (CD)*, 8-11 October 2000, Nashville Tennessee (2000)

[29] Christopher Landauer, Kirstie L. Bellman, "Symbol Systems and Meanings in Virtual Worlds", *Proceedings of VWsim'01: The 2001 Virtual Worlds and Simulation Conference, WMC'2001: The 2001 SCS Western MultiConference*, 7-11 January 2001, Phoenix, SCS (2001)

[30] Christopher Landauer, Kirstie L. Bellman, "Conceptual Modeling Systems: Active Knowledge Processes in Conceptual Categories", pp. 131-144 in Guy W. Mineau (Ed.), *Conceptual Structures: Extracting and Representing Semantics, Contributions to ICCS'2001: The 9th International Conference on Conceptual Structures*, 30 July-03 August 2001, Stanford University (August 2001)

[31] Christopher Landauer, Kirstie L. Bellman, "Intelligent System Architectures with Wrappings", in *Proceedings of CASYS'2001: The Fifth International Conference on Computing Anticipatory Systems*, 13-18 August 2001, Liege, Belgium (2001)

[32] Christopher Landauer, Kirstie L. Bellman, "Architectures for Autonomous Computing Systems", *Proceedings of ANNIE'2001: The 2001 Artificial Neural Nets and Industrial Engineering Conference, Special Track on Bizarre Systems*, 4-7 November 2001, St. Louis, Missouri (2001)

[33] Christopher Landauer, Kirstie L. Bellman, "Abstraction Based Software System Design", *Proceedings of ANNIE'2001: The 2001 Artificial Neural Nets and Industrial Engineering Conference*, 4-7 November 2001, St. Louis, Missouri (2001)

[34] Christopher Landauer, Kirstie L. Bellman, "Computational Infrastructure for Experiments in Cognitive Leverage", in *Proceedings of CT'2001: The Fourth International Conference on Cognitive Technology: Instruments of Mind*, 6-9 August 2001, Warwick, U.K. (2001)

[35] Christopher Landauer, Kirstie L. Bellman, "Self-Modeling Systems", (to appear) in R. Laddaga, H. Shrobe (eds.), "Self-Adaptive Software", Springer Lecture Notes in Computer Science (2002)

[36] Thomas K. Landauer[1], *The Trouble with Computers*, MIT (1995)

[37] Pat Langley, *Elements of Machine Learning*, Morgan-Kaufmann (1996)

[38] Maja J. Mataric, "Behavior-Based Control: Main Properties and Implications", pp. 46-54 in *Proceedings IEEE*

---

[1]no relation

International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems, May 1992, Nice, France (1992)

[39] Maja J. Mataric, "Studying the Role of Embodiment in Cognition", pp. 457-470 in *Cybernetics and Systems*, special issue on *Epistemological Aspects of Embodied Artificial Intelligence*, Volume 28, Number 6 (July 1997)

[40] Alexander M. Meystel, James S. Albus, *Intelligent Systems: Architecture, Design, and Control*, Wiley (2002)

[41] Mary Shaw, William A. Wulf, "Tyrannical Languages *still* Preempt System Design", pp. 200-211 in *Proceedings of ICCL'92: The 1992 International Conference on Computer Languages*, 20-23 April 1992, Oakland, California (1992); includes and comments on Mary Shaw, William A. Wulf, "Toward Relaxing Assumptions in Languages and their Implementations", *ACM SIGPLAN Notices*, Volume 15, No. 3, pp. 45-51 (March 1980)

[42] Lucy A. Suchman, *Plans and Situated Actions: The problem of human-machine communication*, Cambridge U. Press (1987)

[43] Donald O. Walter, Kirstie L. Bellman, "Some Issues in Model Integration", pp. 249-254 in *Proceedings of the SCS Eastern MultiConference*, 23-26 April 1990, Nashville, Tennessee, Simulation Series, Volume 22(3), SCS (1990)